# Nick Murray Capstone 1. Predictive S&P 500 Stock Model

The goal of this project is to establish a predictive model that will determine whether the average price of a given stock will increase or decrease in the next *n* number of days. Accuracy will be tested where n is equal to 5 days, 30 days, and 60 days.

## Scope and Data Collection

Data sourced, cleaned, and packaged through a series of extraction notebooks I created: [Data Extraction Notebooks](https://github.com/nmur1589/CapstoneGit/tree/master/Notebooks/Scraping%20and%20Data%20Extraction):

1. Mass Yahoo Download\_V2: downloads daily high, low, open, close, and volumes for all stocks on the S&P 500 for the past five years
2. Fundamental Scraper: scrapes fundamental company data from MarketWatch using python’s Beautiful Soup package. The scrape includes annual key income and balance sheet metrics for the past five years and quarterly data for the past four quarters
3. Fundamental Calcs: runs a series of conversion functions using regular expressions to clean the scraped fundamental data run a series of financial metric calculations
4. Analyst Scraper: scrapes stock analyst raters from MarketWatch.com (buy, sell, hold, etc) and calculates a “percentage buy” ratio
5. Earnings Calendar: downloads the quarterly earnings release dates for all S&P 500 stocks over the past five years. Creates a field in the dataframe to indicate if the trading day is plus or minus 7 days from a company’s earning release date. (The “Earnings Window” is a particularly volatile time for any given stock. This will be reviewed in more detail in the EDA section)

Note: I packaged most of the functions and routines in the above referenced notebooks into a custom python module called “StockMetrics” to streamline the process moving forward. The StockMetrics module is referenced in the below notebook.

1. Appending Option - Download New Price Data After Mass Download – this provides an easy way to append new daily data to the final data set if I want to update the model with more recent pricing information

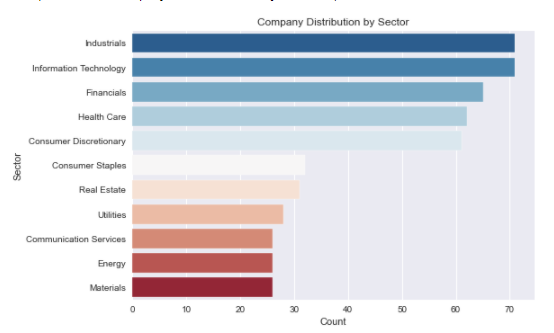
The final dataset produced a pickle file with 566,789 rows and 40 columns that will be explored in the EDA phase

## EDA

Jupyter notebook link: add link to notebook

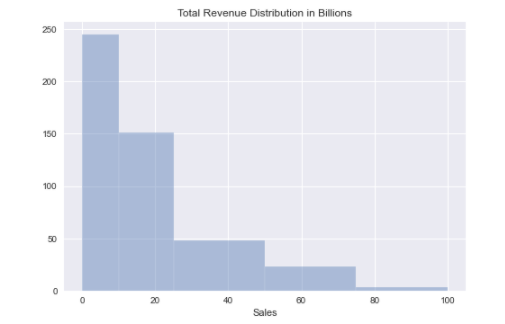
The daily stock detail is comprised of 499 total companies across 11 business sectors segmented as follows:

**Figure 1: Stock symbols by company sector**



**Figure 2: Distribution of Stock Symbols by Annual Sales**

Annual sales range from 0 to $500B/year with most companies falling in the 0 to $25B per year range



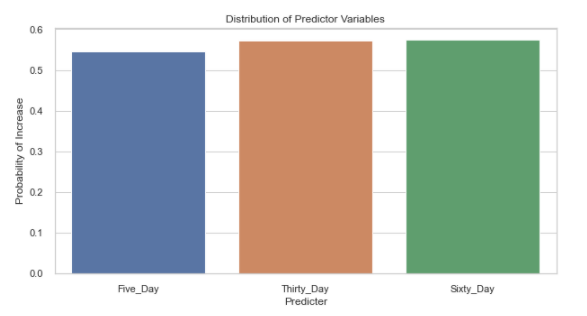
### Summary of X and Y variables

Thee Predictor variables were defined in the data preparation phase. Before the final modeling phase I will be selecting only one to create the prediction model. For the exploratory phase I will be analyzing relationships with all three

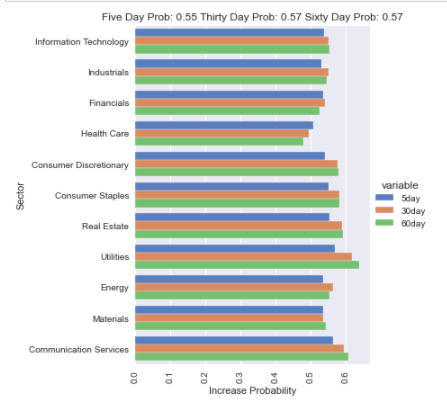
1. **Five Day**: value is equal to 1 if the *average price* increases in the 5 days after the respective market close date; 0 if the price does not increase
2. **Thirty Day**: value is equal to 1 if the *average price* increases in the 30 days after the respective market close date; 0 if the price does not increase
3. **Sixty Day:** value is equal to 1 if the *average price* increases in the 60 days after the respective market close date; 0 if the price does not increase

Below is a summary of the percentage of increases (classification = 1) for all three predictor variables over the entire data set. Given that an increase is classified as 1 while all else is classified as 0 the percentage can also be interpreted **as the probability of increase.** Probabilities of increase range from approximately 55% to 57% as depicted below.

**Figure 3: Distribution of Predictor Variable Percent Increase**

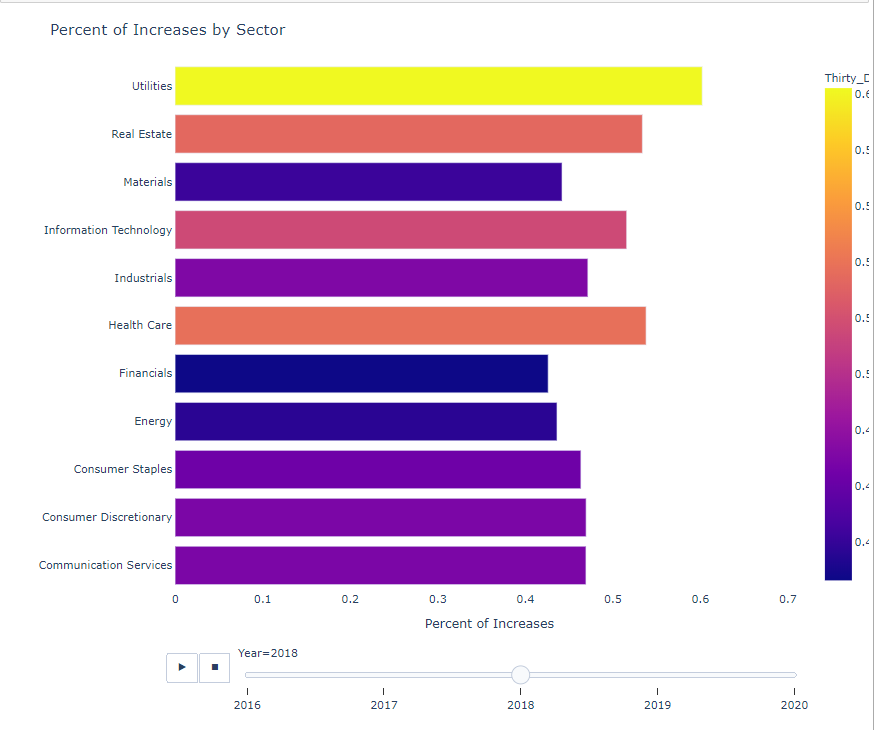


**Figure 4: Distribution of Predictor Variable Percent Increase by Sector**



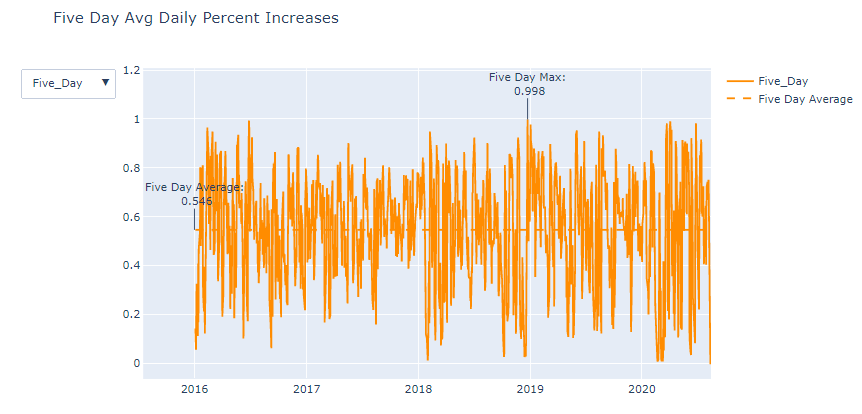
**Figure 5: Figure 4: Distribution of Predictor Variable Percent Increase by Sector with Interactive Year Selection** (Full Functionality in Jupyter Notebook. 2018 Selected as sample below)

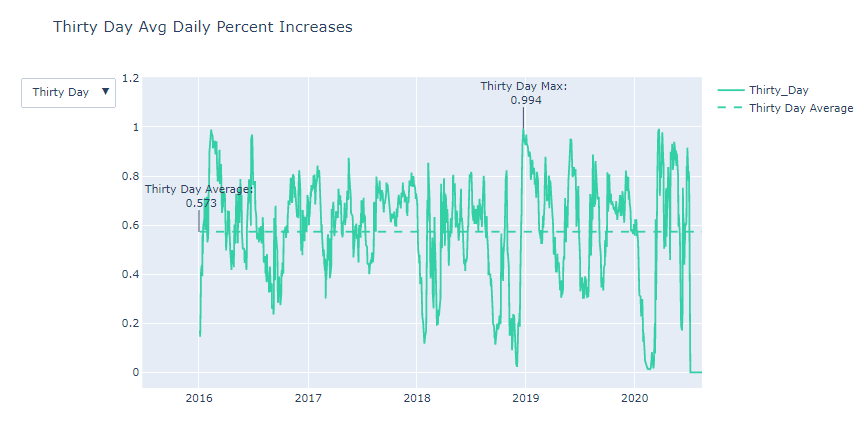
[Click for HTML](C:\\Users\\nmur1\\Google Drive\\Springboard\\Capstone2\\CleanData\\Increases_by_Sector.html)

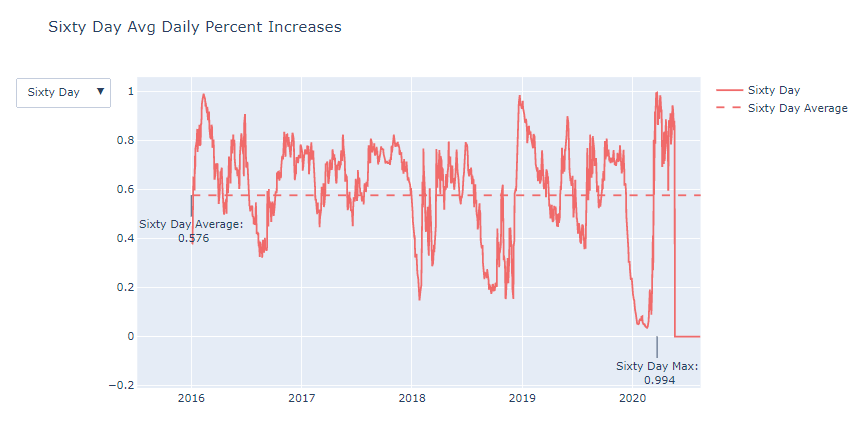


**Figures 6 through 9:** The following plotly graphs show the average daily increases for all stocks in the S&P 500 for the past 5 years. A score of 1.0 indicates that 100% of the stocks increased, 0.0 indicates no stocks increased, .5 indicates that 50% increased, etc. You’ll notice that has the time period we’re analyzing increases from 5 day averages to sixty day the line gets smoother, indicating that the volatility decreases as the time period we’re averaging increases.

[[Click](file:///C:\Users\nmur1\Google%20Drive\Springboard\Capstone2\CleanData\Daily%20Changes.html) for HTML Chart](file:///C:\Users\nmur1\Google%20Drive\Springboard\Capstone2\CleanData\Daily%20Changes.html)

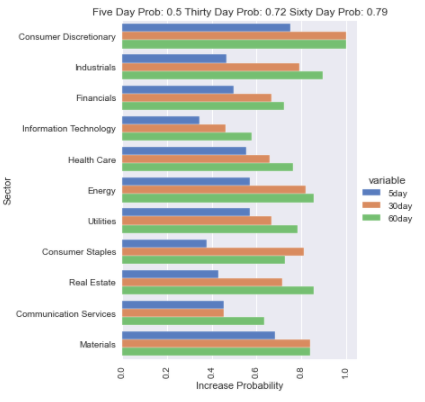






**Figure 10: Highest Probability of Increase:**

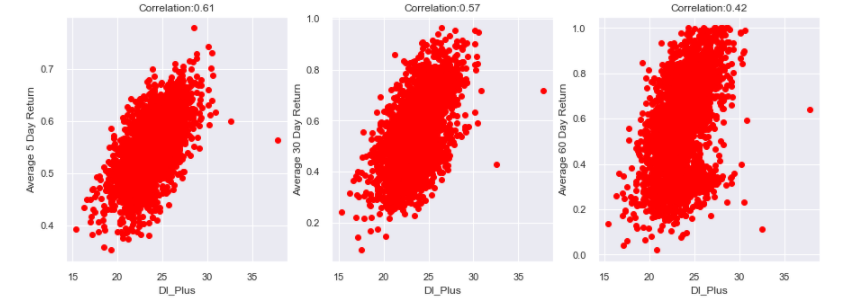
An interesting discovery to note. When the following filters were applied to my dependent variables: *Five Day Exponential Moving Average* is less than the *20 day Simple Moving Average* and the *14 day slope of DI Plus* times the *14 day r squared of DI Plus* is greater than 1, the 30 and 60 day probabilities for increasing dramatically increase. Although this phenomenon only happened 292 times out of 500,000 observations.



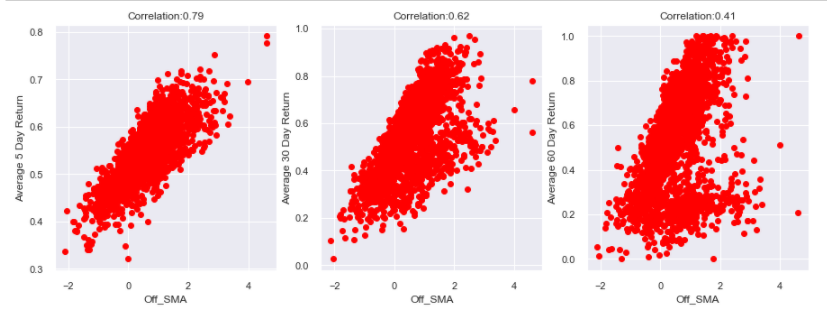
## Correlation Analysis

The following scatter plots analyze the correlation of various feature variables to the three predictor variables (Average 5 day increase, Average 30 Day Increase, Average 60 Day increase). All the below variables appear to have a relatively high correlation to the 5 day and 30 day future averages while future 60 day averages offer mixed results

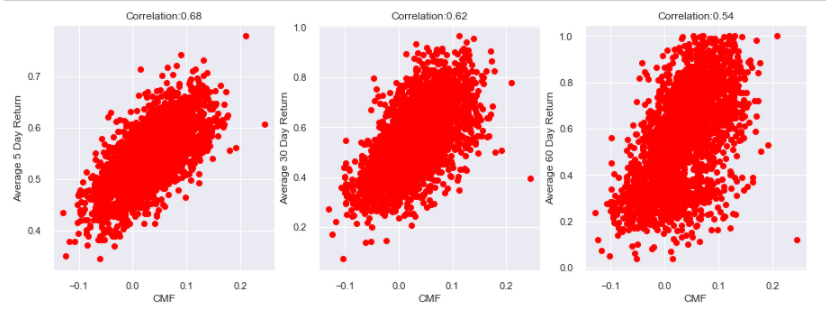
**X Variable: Directional Index Plus correlation**



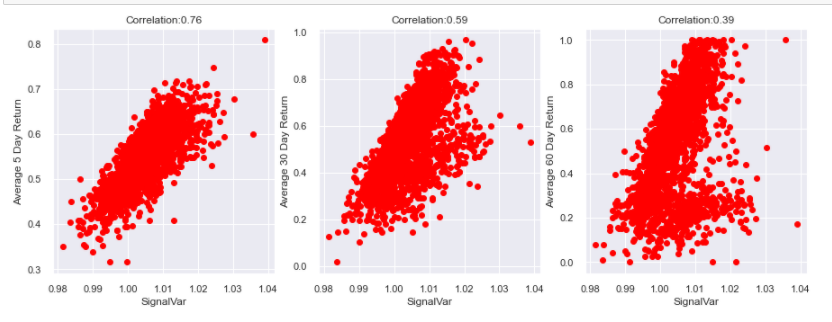
**X Variable: Percentage off of 20-day simple moving average**



**X Variable: Chaikin Money Flow**

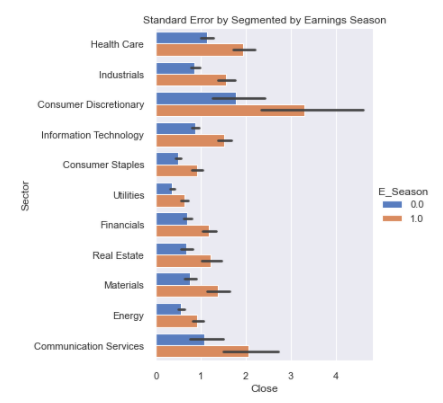


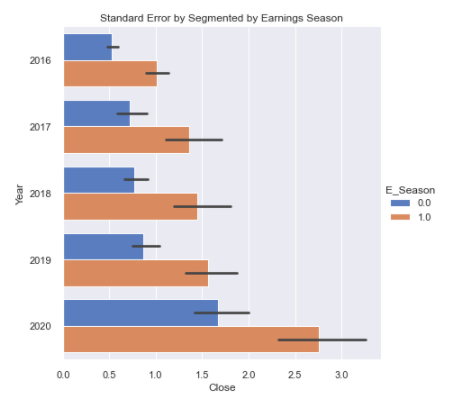
**X Variable: Signal Variance (5 Day Exponential Moving Average divided by 20 Day Simple Moving Average)**

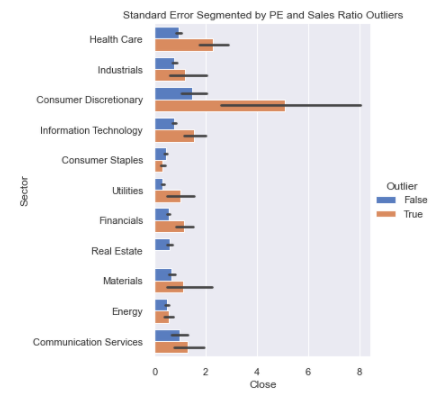


## Earnings Season Volatility

One of the many challenges in predictive analysis algorithms in the stock market is that a positive or negative earnings report can dramatically alter the prior and future patterns of the stock price as they relate to past and future technical indicators – or X variables. Using python’s yfinance package I was able to import the earnings release date for all 499 companies in scope for the past five years and define an “Earnings Season” as one of my X variables. If the trading day falls within plus or minus seven days of the earnings release date it is flagged as a “1”. The below charts illustrate the difference in the average stock prices’ standard error in and outside of the earning’s season. The blue bars represent trading days not in season while the orange bars are in season. From 2016 to 2020 the standard error for stock prices trading outside of the earnings window was .8897 versus 1.7891 inside the earnings window. Due to the volatile nature of pricing during this time period the final model will filter out days that are trading during this time frame. For future application of the model, stocks trading within their respective earnings window will not be candidates for the predictive model either.





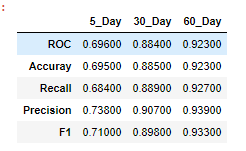


## Initial Accuracy and Preprocessing

Before moving to my pre-processing and modeling phase I’d like to pick only one predictor variable to test and create the final model (5-Day Average Increase, 30-Day Average Increase, or 60-Day Average Increase)

As a final step to the EDA I ran all three scenarios through a RandomForest Classifier to get a high level idea on which group could yield the most accurate results.

Summary scores are listed below. Moving forward I’m going to choose the 30\_Day average as my predictor y variable. 5\_Day averages do not appear to provide consistent average results. While 60\_Day is showing the highest accuracy, my correlation analysis in the earlier EDA stage indicated that the dependent variables had a better correlation to the y variable.



## Pre-processing and Feature Selection

Link to notebook:

### **Model Selection**

After training and scaling the data 4 classifier models were initially run to determine the best model to proceed with. Per the below results **Random Forest** was selected:

decision tree: 0.8464

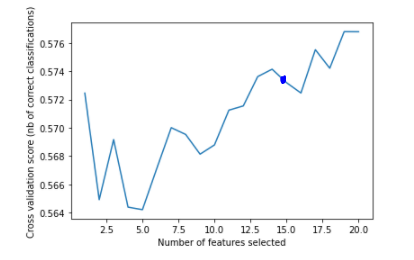
random forest: 0.9197

Ada Boost: 0.5847

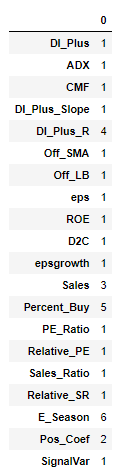
Gradient Boosting: 0.589

### **Feature Selection**

Sklearn’s RFECV module has determined that 20 features should produce the most accurate results. However, as I want to balance the bias variance trade-off, I settled on 15 features noted by the blue marker in the below graph.



Running recursive feature elimination with 15 features determines that I should drop the [DI\_Plus\_R, Sales, Percent\_Buy, E\_Season, and Pos\_Coef] variables from my set, keeping everything that scores a 1 in the below results



### Hyper Parameter Tuning

Running a Randomized Grid Search with the following options I determined that the best parameters were to keep the default settings of the RandomForest model with the exception of the bootstrap argument which I will set to False



**Best Params**

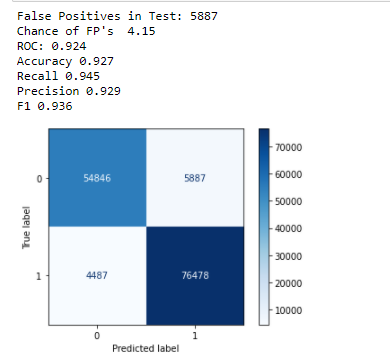
100 estimators, 2 min samples split, 1 min samples leaf, bootstrap = false

Best Score = 90.3

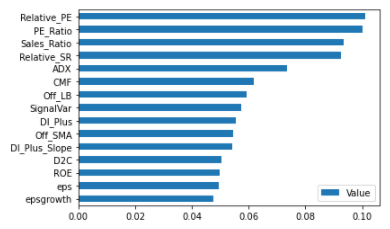
### Scoring using Train\_Test\_Split

Initial scoring on a 75/25 train/test split yielded promising results outlined in the below confusion matrix:

**Model Score**



**Feature Importance**



With a 92.7% accuracy rate, 92.9% precision score, and only 4.15% of false positives, this is starting to look like an impressive stock prediction model!

However we need to ensure that the model is not overfitting the training data and can properly generalize on “unseen” data (bias variance tradeoff).

To further assess accuracy, I will run a 3 fold cross validation test on the dataset.

Unfortunately, this test produced results far less accurate than a standard train test split:

Results: [0.58487633, 0.56147397, 0.53581226].

**Final Accuracy Results**

Using a grid search wrapper technique obtained from my mentor, Max Sop, I was able to return scores back to their original metrics of greater than 90%

0.9235896417978543

Best params for roc\_auc\_score

Confusion matrix of Random Forest optimized for roc\_auc\_score on the test data:

pred\_neg pred\_pos

neg 54804 5929

pos 4469 76496

precision recall f1-score support

0 0.92 0.90 0.91 60733

1 0.93 0.94 0.94 80965

accuracy 0.93 141698

macro avg 0.93 0.92 0.92 141698

weighted avg 0.93 0.93 0.93 141698

**Model Production**

Production Notebook Link:

Using the model defined through the above process I will run unseen data with stock metrics and prices as of 08/27.

My vision is to publish the results to a tableau workspace

[Model Predictions](https://public.tableau.com/profile/nicholas.murray#!/vizhome/SpringboardCapstone_Predictions/30DayPredictions)

[Supporting Prices and Technical Indicators](https://public.tableau.com/profile/nicholas.murray#!/vizhome/SpringboardCapstone_15987427385890/TechIndicators)